```
RRRRRRRRRRRR     MMM           MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM           MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM           MMM     SSSSSSSSSSSS
RRR        RRR   MMMMMM     MMMMMM     SSS
RRR        RRR   MMMMMM     MMMMMM     SSS
RRR        RRR   MMMMMM     MMMMMM     SSS
RRR        RRR   MMM   MMM   MMM       SSS
RRR        RRR   MMM     MMM   MMM     SSS
RRRRRRRRRRRR     MMM           MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM           MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM           MMM     SSSSSSSSS
RRR    RRR       MMM           MMM            SSS
RRR    RRR       MMM           MMM            SSS
RRR    RRR       MMM           MMM            SSS
RRR      RRR     MMM           MMM            SSS
RRR       RRR    MMM           MMM            SSS
RRR       RRR    MMM           MMM            SSS
RRR         RRR  MMM           MMM     SSSSSSSSSSSS
RRR         RRR  MMM           MMM     SSSSSSSSSSSS
RRR         RRR  MMM           MMM     SSSSSSSSSSSS
```

RM2GET
LIS

RM2GET
V04-001

RELATIVE SPECIFIC GET AND FIND   J 4   16-SEP-1984 01:03:37 VAX/VMS Macro V04-00   Page   1
                                                    7-SEP-1984 17:13:37 [RMS.SRC]RM2GET.MAR;2   (1)

```
0000       1                  $BEGIN  RM2GET,001,RM$RMS2,<RELATIVE SPECIFIC GET AND FIND>
0000       2          ;
0000       3          ;
0000       4          ;********************************************************************
0000       5          ;*                                                                  *
0000       6          ;*   COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                        *
0000       7          ;*   DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.         *
0000       8          ;*   ALL RIGHTS RESERVED.                                           *
0000       9          ;*                                                                  *
0000      10          ;*   THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
0000      11          ;*   ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE *
0000      12          ;*   INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER *
0000      13          ;*   COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
0000      14          ;*   OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY *
0000      15          ;*   TRANSFERRED.                                                    *
0000      16          ;*                                                                  *
0000      17          ;*   THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE *
0000      18          ;*   AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT *
0000      19          ;*   CORPORATION.                                                    *
0000      20          ;*                                                                  *
0000      21          ;*   DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS *
0000      22          ;*   SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.         *
0000      23          ;*                                                                  *
0000      24          ;*                                                                  *
0000      25          ;********************************************************************
0000      26          ;
```

```
0000    28  ;++
0000    29  ; Facility: rms32
0000    30  ;
0000    31  ; Abstract:
0000    32  ;            this module provides relative file organization-
0000    33  ;            specific processing for the $get and $find functions.
0000    34  ;
0000    35  ; Environment:
0000    36  ;            star processor running starlet exec.
0000    37  ;
0000    38  ; Author:      L F Laverdure,  creation date: 4-NOV-1977
0000    39  ;
0000    40  ; Modified By:
0000    41  ;
0000    42  ;     V04-001  JEJ0054          James E Johnson          07-Sep-1984
0000    43  ;              Fix error handling in a $PUT operation to not try to
0000    44  ;              release a BDB that we don't have.
0000    45  ;
0000    46  ;     V03-006  DGB0065          Donald G. Blair          02-Jul-1984
0000    47  ;              Fix error handling to report rms$_key correctly when
0000    48  ;              user tries to $FIND an illegal relative record number
0000    49  ;              on a shared relative file.
0000    50  ;
0000    51  ;     V03-005  KBT0447          Keith B. Thompson        5-Dec-1982
0000    52  ;              Change ref. rm$cachec to rm$cache
0000    53  ;
0000    54  ;     V03-004  KBT0317          Keith B. Thompson        8-Sep-1982
0000    55  ;              Remove all SO sharing code
0000    56  ;
0000    57  ;     V03-003  KBT0128          Keith B. Thompson        19-Aug-1982
0000    58  ;              Remove a ref. to set_sifb_adr i forgot in KBT0114 and
0000    59  ;              reorganize psects
0000    60  ;
0000    61  ;     V03-002  KPL0002          Peter Lieberwirth        19-Aug-1982
0000    62  ;              Fix bug where shared access of record past MRN (should
0000    63  ;              be illegal) returned status OK_ALK.  This was due to a
0000    64  ;              failure to check for error on return from GETFIND2.
0000    65  ;
0000    66  ;     V03-001  KBT0114          Keith B. Thompson        6-Aug-1982
0000    67  ;              Remove ref to set_sifb_adr
0000    68  ;
0000    69  ;     V02-032  RAS0063          Ron Schaefer             29-Jan-1982
0000    70  ;              Correct probes of the user's key and record buffers.
0000    71  ;
0000    72  ;     V02-031  CDS0002          C Saether                15-Dec-1981
0000    73  ;              Load R4 with CURBDB at SETR7 point such that it reflects
0000    74  ;              the fact that some errors from RM$LOCK will leave the
0000    75  ;              current bucket not accessed.
0000    76  ;
0000    77  ;     V02-030  CDS0001          C Saether                10-Dec-1981
0000    78  ;              Fix broken branch.
0000    79  ;
0000    80  ;     V02-029  KPL0001          Peter Lieberwirth        9-Jul-1981
0000    81  ;              Add support for new record locking functions: lock for READ,
0000    82  ;              and WAIT on record lock conflict.  WAIT requires reaccessing
0000    83  ;              bucket after successful wait for record lock.
0000    84  ;
```

```
0000    85 ;                    Also clean up some commentary that was incorrectly reformatted.
0000    86 ;
0000    87 ;        V02-028 CDS0075         C D Saether       28-Aug-1980       16:20
0000    88 ;                    Make check for MRS correctly with VFC format records.
0000    89 ;
0000    90 ;--
0000    91 ;
0000    92
```

RM2GET
V04-001

M 4

RELATIVE SPECIFIC GET AND FIND
DECLARATIONS

16-SEP-1984 01:03:37 VAX/VMS Macro V04-00
7-SEP-1984 17:13:37 [RMS.SRC]RM2GET.MAR;2

Page 4
(3)

```
                    0000      94                .SBTTL  DECLARATIONS
                    0000      95
                    0000      96        ;
                    0000      97        ; Include Files:
                    0000      98        ;
                    0000      99
                    0000     100        ;
                    0000     101        ; Macros:
                    0000     102        ;
                    0000     103
                    0000     104                $IRBDEF
                    0000     105                $IFBDEF
                    0000     106                $RABDEF
                    0000     107                $FABDEF
                    0000     108                $DLCDEF
                    0000     109                $BDBDEF
                    0000     110                $CSHDEF
                    0000     111                $RMSDEF
                    0000     112
                    0000     113
                    0000     114        ; Equated Symbols:
                    0000     115        ;
                    0000     116
        00000020    0000     117                ROP=RAB$L_ROP*8                  ; bit offset to rop
                    0000     118
                    0000     119
                    0000     120        ; Own Storage:
                    0000     121        ;
                    0000     122
```

RM2GET
V04-001

RELATIVE SPECIFIC GET AND FIND   16-SEP-1984 01:03:37 VAX/VMS Macro V04-00   Page 5
RM$GET2/RM$FIND2 - REL. $GET & $FIND   7-SEP-1984 17:13:37 [RMS.SRC]RM2GET.MAR;2   (4)

```
                        0000  124              .SBTTL  RM$GET2/RM$FIND2 - REL. $GET & $FIND
                        0000  125
                        0000  126  ;++
                        0000  127  ; RM$CLN2_PUT
                        0000  128  ; RM$CLN2_UPD
                        0000  129  ; RM$CLN2_DEL
                        0000  130  ; RM$GET2
                        0000  131  ; RM$RLS2
                        0000  132  ; RM$FIND2
                        0000  133  ;
                        0000  134  ; this module performs the following functions:
                        0000  135  ;
                        0000  136  ;         1. common $get/$find setup
                        0000  137  ;         2. accesses the bucket, locks the record if necessary, and
                        0000  138  ;            for get, copies the record to the user buffer if move mode,
                        0000  139  ;            setting the various rab fields as required.
                        0000  140  ;         3. set "last-operation-was-a-find" and nrp context
                        0000  141  ;
                        0000  142  ;
                        0000  143  ; Calling sequence:
                        0000  144  ;
                        0000  145  ;         entered via case branch from rms$get
                        0000  146  ;         or rms$find at rm$get2 or rm$find2 respectively.
                        0000  147  ;
                        0000  148  ;         exit is to user via rm$exrms.
                        0000  149  ;
                        0000  150  ;
                        0000  151  ; Input Parameters:
                        0000  152  ;
                        0000  153  ;         r11       impure area address
                        0000  154  ;         r10       ifab address
                        0000  155  ;         r9        irab address
                        0000  156  ;         r8        rab address
                        0000  157  ;
                        0000  158  ;
                        0000  159  ; Implicit Inputs:
                        0000  160  ;
                        0000  161  ;         the contents of the rab and related irab and ifab.
                        0000  162  ;         in particular, irb$v_find must be set if doing $find, else clear.
                        0000  163  ;
                        0000  164  ; Output Parameters:
                        0000  165  ;
                        0000  166  ;         r7 - r1           destroyed
                        0000  167  ;         r0                status
                        0000  168  ;
                        0000  169  ;
                        0000  170  ; Implicit Outputs:
                        0000  171  ;
                        0000  172  ;         various fields of the rab are filled in to reflect the status of
                        0000  173  ;         the operation (see functional spec for details).
                        0000  174  ;
                        0000  175  ;         the irab is similarly updated.
                        0000  176  ;
                        0000  177  ;
                        0000  178  ; Completion Codes:
                        0000  179  ;
                        0000  180  ;         standard rms (see functional spec).
```

RM2GET
V04-001
RELATIVE SPECIFIC GET AND FIND
RMSGET2/RMSFIND2 - REL. $GET & $FIND
B 5
16-SEP-1984 01:03:37   VAX/VMS Macro V04-00
7-SEP-1984 17:13:37   [RMS.SRC]RM2GET.MAR;2
Page   6
(4)

```
0000   181 ;
0000   182 ; Side Effects:
0000   183 ;
0000   184 ;--     none
0000   185 ;--
0000   186
```

```
                           0000  188                    .SBTTL  CLEANUP CODE
                           0000  189
                           0000  190  ;
                           0000  191  ;  code to clean up on errors (note: this is not the entry point to rm$get2)
                           0000  192  ;
                           0000  193  ;  there are various entry points for the cleanup depending upon the function
                           0000  194  ;  being executed.  zeroes the rsz, bkt, and rfa fields of the rab, unlocks
                           0000  195  ;  the rp and resets various irab flags, releases the bucket, and exits rms.
                           0000  196  ;
                           0000  197  ;  inputs:
                           0000  198  ;       r7         status code
                           0000  199  ;       r4         bdb address or 0 if none
                           0000  200  ;
                           0000  201
                           0000  202  CLEANUP:
         22 A8   B4        0000  203                    CLRW     RAB$W_RSZ(R8)              ; indicate no record
                           0003  204
                           0003  205  ;
                           0003  206  ;  entry point for $find and $put
                           0003  207  ;
                           0003  208
                           0003  209  RM$CLN2_PUT::
                           0003  210                    ASSUME   RAB$C_SEQ EQ 0
         1E A8   95        0003  211                    TSTB     RAB$B_RAC(R8)              ; seq. access?
            03   12        0006  212                    BNEQ     RM$CLN2_UPD               ; branch if not
         38 A8   D4        0008  213                    CLRL     RAB$L_BKT(R8)             ; clear the record #
                           000B  214
                           000B  215  ;
                           000B  216  ;  entry point for $update
                           000B  217  ;
                           000B  218
                           000B  219  RM$CLN2_UPD::
         10 A8   D4        000B  220                    CLRL     RAB$W_RFA(R8)             ; zero the rfa
         14 A8   B4        000E  221                    CLRW     RAB$W_RFA+4(R8)           ;      ''
                           0011  222
                           0011  223  ;
                           0011  224  ;  entry point for $delete
                           0011  225  ;
                           0011  226
                           0011  227  RM$CLN2_DEL::
   0B 69   2D   E5        0011  228                    BBCC     #IRB$V_UNLOCK_RP,(R9),5$; don't unlock if manual lock
   51 48 A9   D0          0015  229                    MOVL     IRB$L_RP(R9),R1           ; get rp
            08   13        0019  230                    BEQL     10$                       ; branch if none
            52   D4        001B  231                    CLRL     R2                        ; high order lock value
         FFE0'  30        001D  232                    BSBW     RM$UNLOCK                  ; unlock record if locked (ignore error)
         48 A9   D4        0020  233  5$:               CLRL     IRB$L_RP(R9)              ; show no current record
                           0023  234  10$:              CSB      #IRB$V_FIND_LAST,(R9)     ; clear find last
         00FA   31        0027  235                    BRW      RM$RLS2                    ; go release bucket
                           002A  236
```

```
                          002A   238                    .SBTTL  $GET CODE
                          002A   239
                          002A   240          ;
                          002A   241          ;  entry point for relative-specific get
                          002A   242          ;
                          002A   243
                          002A   244          RM$GET2::
                          002A   245                    STSTPT  GET2
            015E   30     0030   246                    BSBW    GETREC2                     ; go access bucket
            0279   30     0033   247                    BSBW    GETFIND2                    ; go access record
            C7 57   E9    0036   248                    BLBC    R7,CLEANUP                  ; get out on error
            0318   30     0039   249                    BSBW    GETLOCK2                    ; lock the record, if necessary
            C1 57   E9    003C   250          NTRETG:   BLBC    R7,CLEANUP                  ; get out on error
                          003F   251                    CSB     #IRB$V_FIND_LAST,(R9)       ; clear find last flag
      8049 8F   57  B1    0043   252                    CMPW    R7,#RMS$_OK_RNF&^XFFFF      ; was status = record not found?
                 46  13   0048   253                    BEQL    NULL_REC                    ; branch if yes
                          004A   254
                          004A   255          ;
                          004A   256          ;  set the rab$w_rsz field based upon the record format
                          004A   257          ;
                          004A   258
          01   50 AA  91  004A   259                    CMPB    IFB$B_RFMORG(R10),#FAB$C_FIX ; rfm=fix?
                 03  12   004E   260                    BNEQ    10$
                 0086 31  0050   261                    BRW     RSZFIX                      ; branch if yes
             56   85  3C  0053   262          10$:      MOVZWL  (R5)+,R6                    ; set variable record length
          03   50 AA  91  0056   263                    CMPB    IFB$B_RFMORG(R10), #FAB$C_VFC ; VFC record format?
                 08  13   005A   264                    BEQL    VFCREC                      ; branch if vfc
          60 AA   56  B1  005C   265                    CMPW    R6,IFB$W_MRS(R10)           ; size within range?
                 7B  1B   0060   266                    BLEQU   CHKLOC                      ; size ok, br and continue
                 5B  11   0062   267                    BRB     ERRIRC                      ; size illegal, br to error
                          0064   268
                          0064   269          ;
                          0064   270          ;  VFC record format.
                          0064   271          ;
                          0064   272          ;  Adjust record size for fixed header size. Check against MRS to see if legal.
                          0064   273          ;  Move the header to the record header buffer.
                          0064   274          ;
                          0064   275
                          0064   276          VFCREC:
         50   5F AA  9A   0064   277                    MOVZBL  IFB$B_FSZ(R10),R0           ; pick up header size
             56   50  A2  0068   278                    SUBW2   R0,R6                       ; adjust record length
          60 AA   56  B1  006B   279                    CMPW    R6, IFB$W_MRS(R10)          ; record size within bounds?
                 4E  1A   006F   280                    BGTRU   ERRIRC                      ; gtru nope. report error.
          51   2C A8  D0  0071   281                    MOVL    RAB$L_RHB(R8),R1            ; get rhb address
                 14  13   0075   282                    BEQL    10$                         ; and branch if none
                          0077   283                    IFNOWRT R0,(R1),ERRRHB,IRB$B_MODE(R9); branch if not writable
                 54  DD   007E   284                    PUSHL   R4                          ; save bdb address
       61   65  50  28    0080   285                    MOVC3   R0,(R5),(R1)                ; move the record header
                 10  BA   0084   286                    POPR    #^M<R4>                     ; restore bdb address
             55   51  D0  0086   287                    MOVL    R1,R5                       ; update record buffer addr
                 03  11   0089   288                    BRB     20$
             55   50  C0  008B   289          10$:      ADDL2   R0,R5                       ; skip unwanted header
                 4D  11   008E   290          20$:      BRB     CHKLOC                      ; go pick up main sequence
                          0090   291
```

```
                            0090    293
                            0090    294  ;
                            0090    295  ;    status from getfind2 was ok_rnf.
                            0090    296  ;    this implies user has specified read of a non-existent record.
                            0090    297  ;
                            0090    298  ;
                            0090    299  NULL_REC:
          56      B4        0090    300          CLRW      R6                          ; show zero len
    03 50 AA      B1        0092    301          CMPW      IFBSB_RFMORG(R10),#FABSC_VFC; rfm = vfc?
          16      12        0096    302          BNEQ      20$                         ; branch if not
    51 2C AB      D0        0098    303          MOVL      RABSL_RHB(R8),R1            ; rhb specified?
          10      13        009C    304          BEQL      20$                         ; branch if not
    50 5F AA      9A        009E    305          MOVZBL    IFBSB_FSZ(R10),R0          ; get header len
                            00A2    306          IFNOWRT   R0,(R1),ERRRHB,IRBSB_MODE(R9); branch if not writable
          81      94        00A9    307  10$:     CLRB      (R1)+                       ; clear the buffer
       FB 50      F5        00AB    308          SOBGTR    R0,10$                      ; loop
          6A      11        00AE    309  20$:     BRB       SETRSZ                      ; go finish up
                            00B0    310
                            00B0    311
                            00B0    312  ERRRHB:  RMSERR    RHB,R7                      ; bad record header buffer
                            00B5    313
                            00B5    314  ;
                            00B5    315  ;    note: all errors detected after successfully returning from getfind2
                            00B5    316  ;    must exit thru here to determine whether the current record must be
                            00B5    317  ;    unlocked on errors.  see notes in locking code in getfind2.
                            00B5    318  ;
                            00B5    319
                            00B5    320  CLN1BR:
       04 5C      E9        00B5    321          BLBC      AP,10$                      ; lbc no special action to
                            00B8    322                                                ; unlock rp on error
                            00B8    323          SSB       #IRBSV_UNLOCK_RP,(R9)       ; make sure record is unlocked
       FF41       31        00BC    324  10$:     BRW       CLEANUP                     ; go clean up
                            00BF    325
                            00BF    326  ;
                            00BF    327  ;    handle errors
                            00BF    328  ;
                            00BF    329
                            00BF    330  ERRIRC:
                            00C4    331          RMSERR    IRC,R7                      ; illegal record size in file
    0C AB 38 AB   D0        00C4    332          MOVL      RABSL_BKT(R8),RABSL_STV(R8); indicate rrn of bad record
          EA      11        00C9    333          BRB       CLN1BR                      ; go clean up
                            00CB    334
                            00CB    335  ERRUSZ:
                            00CB    336          RMSERR    USZ,R7                      ; 0 user buffer len
          E3      11        00D0    337          BRB       CLN1BR                      ; go clean up
                            00D2    338
                            00D2    339  ERRUBF:
                            00D2    340          RMSERR    UBF,R7                      ; invalid user buffer
          DC      11        00D7    341          BRB       CLN1BR                      ; go clean up
                            00D9    342
```

```
                              00D9   344   ;   set record size from mrs for fixed length record format
                              00D9   345   ;
                              00D9   346   ;
                              00D9   347   ;
        56    60 AA   B0      00D9   348   RSZFIX:  MOVW     IFB$W_MRS(R10),R6
                              00DD   349
                              00DD   350   ;
                              00DD   351   ;   if locate mode asked for and allowable, return pointer to record,
                              00DD   352   ;   else copy record to user buffer.
                              00DD   353   ;
                              00DD   354
        10 68    30   E1      00DD   355   CHKLOC:  BBC      #RAB$V_LOC+ROP,(R8),MOVE_MODE; branch if locate mode not speced
  0B 22 AA    03   E0         00E1   356            BBS      #FAB$V_UPD,IFB$B_FAC(R10),MOVE_MODE; or if update accessed
  06 0A A4    04   E0         00E6   357            BBS      #BDB$V_NOLOCATE,BDB$B_FLGS(R4),MOVE_MODE; or if bdb says no
        28 A8    55   D0      00EB   358            MOVL     R5,RAB$L_RBF(R8)           ; set rbf from record address
              29   11         00EF   359            BRB      SETRSZ                     ; go set record size
                              00F1   360
                              00F1   361   ;
                              00F1   362   ;   move mode
                              00F1   363   ;
                              00F1   364   ;   check out the user buffer and copy the record.
                              00F1   365   ;
                              00F1   366
                              00F1   367   MOVE_MODE:
        50    20 A8   3C      00F1   368            MOVZWL   RAB$W_USZ(R8),R0          ; get user buffer size
              D4   13         00F5   369            BEQL     ERRUSZ                    ; error if none
        56    50   B1         00F7   370            CMPW     R0,R6                     ; usz < rsz?
              64   1F         00FA   371            BLSSU    ERRRTB                    ; branch if yes
     53   24 B8   DE          00FC   372   PROBEB:  MOVAL    @RAB$L_UBF(R8),R3         ; get buffer addr
  0200 8F    56   B1          0100   373            CMPW     R6,#512                   ; record greater than 2 pages?
           36   1A            0105   374            BGTRU    LONG_PROBE                ; branch if yes
                              0107   375            IFNOWRT  R6,(R3),ERRUBF,IRB$B_MODE(R9); branch if ubf not writable
           54   DD            010E   376   MOVREC:  PUSHL    R4                        ; save bdb address
     28 A8    53   D0         0110   377            MOVL     R3,RAB$L_RBF(R8)          ; set record address
  63   65    56   28          0114   378            MOVC3    R6,(R5),(R3)              ; copy record
              10   BA         0118   379            POPR     #^M<R4>                   ; restore bdb address
     22 A8    56   B0         011A   380   SETRSZ:  MOVW     R6,RAB$W_RSZ(R8)          ; and set record size
  40 A9    01   48 A9   C1    011E   381   SETNRP:  ADDL3    IRB$L_RP(R9),#1,IRB$L_NRP(R9); set nrp from rp+1
                              0124   382
                              0124   383   ;
                              0124   384   ;   release access to the bucket.
                              0124   385   ;   will cause write to occur if dirty and deferred write not set.
                              0124   386   ;
                              0124   387
                              0124   388   RM$RLS2::  TSTL   R4                        ; is there a bdb?
           54   D5            0124   389            TSTL     R4                        ; is there a bdb?
           0B   13            0126   390            BEQL     10$                       ; branch if none
           53   D4            0128   391            CLRL     R3                        ; no options wanted
        FED3'  30             012A   392            BSBW     RMSRELEASE                ; release access to bucket
        03 57   E9            012D   393            BLBC     R7,10$                    ; branch if already had error
        03 50   E9            0130   394            BLBC     R0,20$                    ; branch if release failed
        50   57   D0          0133   395   10$:     MOVL     R7,R0                     ; status to r0
              0136             0136   396   20$:     CSB      #IRB$V_FIND,(R9)          ; clear 'doing find'
        FEC3'  31             013A   397            BRW      RMSEXRMS                  ; exit rms
```

```
013D      399
013D      400  ;
013D      401  ;   probe writeablity of all pages ( > 1) of user buffer
013D      402  ;
013D      403
013D      404  LONG_PROBE:
      50    56    D0    013D      405          MOVL    R6,R0                      ; copy buffer length
      51    53    D0    0140      406          MOVL    R3,R1                      ; and address
52    FE00  8F    32    0143      407          CVTWL   #-512,R2                   ; set address constant
                       0148      408  10$:     IFNOWRT R0,(R1),ERRUBF,IRB$B_MODE(R9); branch if not writable
      51    52    C2    014F      409          SUBL2   R2,R1                      ; get address next page
   50    6042    3E    0152      410          MOVAW   (R0)[R2],R0                ; calculate new length
         F0    14    0156      411          BGTR    10$                        ; branch if more to probe
      50    52    C2    0158      412          SUBL2   R2,R0                      ; final page to probe?
         EB    14    015B      413          BGTR    10$                        ; branch if yes
      FFAE    31    015D      414          BRW     MOVREC                     ; return to main sequence
                  0160      415
                  0160      416  ;
                  0160      417  ;   record too long for user buffer.
                  0160      418  ;   note error and actual length and adjust count to fill user buffer.
                  0160      419  ;
                  0160      420
                  0160      421  ERRRTB:
                  0160      422          RMSERR  RTB,R7                     ; show record too big error
0C A8    56    D0    0165      423          MOVL    R6,RAB$L_STV(R8)           ; tell user actual length
      56    50    D0    0169      424          MOVL    R0,R6                      ; but copy only usz amount
      FF8D    31    016C      425          BRW     PROBEB                     ; return to main sequence
                  016F      426
```

```
                      016F      428              .SBTTL  $FIND CODE
                      016F      429
                      016F      430      ;
                      016F      431      ;   entry point for $find function.
                      016F      432      ;
                      016F      433      ;
                      016F      434      RM$FIND2::
                      016F      435              $TSTPT  FIND2
           1A    10   0175      436              BSBB    GETREC2                 ; go access bucket
         0135    30   0177      437              BSBW    GETFIND2                ; go access record
        03 57    E9   017A      438              BLBC    R7,NTRETF               ; br on error
         01D4    30   017D      439              BSBW    GETLOCK2                ; lock the record, if necessary
                      0180      440
                      0180      441      NTRETF: SSB     #IRB$V_FIND_LAST,(R9)   ; set last opr. was a find
        07 57    E9   0184      442              BLBC    R7,10$                  ; branch on error
                      0187      443              ASSUME  RAB$C_SEQ EQ 0
        1E A8    95   0187      444              TSTB    RAB$B_RAC(R8)           ; sequential access?
           98    12   018A      445              BNEQ    RM$RL52                 ; branch if not
           90    11   018C      446              BRB     SETNRP                  ; yes - set nrp
         FE72    31   018E      447      10$:     BRW     RM$CLN2_PUT            ; clean up on error
```

I   5

```
0191   449                    .SBTTL   GETREC2 - ROUTINE TO LOCATE RECORD IN BUFFER
0191   450
0191   451    ;++
0191   452    ; RM$GETREC2_PUT
0191   453    ; RM$READBKT2_UPD
0191   454    ; RM$READBKT2
0191   455    ;
0191   456    ;   this routine performs the following functions:
0191   457    ;
0191   458    ;       1.  sets record number from rp, nrp, kbf or rfa depending upon
0191   459    ;           rab$b_rac and irb$v_find_last
0191   460    ;       2.  if irb$v_unlock_rp set, unlocks record specified by rp on entry
0191   461    ;           rp will not be unlocked if this is a sequential get following a
0191   462    ;           find or if this is a random access for the same record as the
0191   463    ;           current record.  this eliminates a window where the record lock
0191   464    ;           could be lost accessing the same record.
0191   465    ;       3.  checks for record number valid
0191   466    ;       4.  calculates vbn and offset for record
0191   467    ;       5.  if bucket not past eof, calls rm$cache to locate the record
0191   468    ;           (possibly reading it in)
0191   469    ;       6.  calculates address of record in buffer
0191   470    ;
0191   471    ;   calling sequence:
0191   472    ;
0191   473    ;       bsbw    getrec2
0191   474    ;
0191   475    ;       alternate entry at rm$getrec2_put for $put
0191   476    ;
0191   477    ;   input parameters:
0191   478    ;
0191   479    ;       r11         impurea area address
0191   480    ;       r10         ifab address
0191   481    ;       r9          irab address
0191   482    ;       r8          rab address
0191   483    ;       r3          csh$m_lock flag if entry at rm$getrec2_put
0191   484    ;
0191   485    ;   implicit inputs:
0191   486    ;
0191   487    ;       rab$b_rac           record access mode
0191   488    ;       rab$w_rfa           if rac = rfa
0191   489    ;       rab$l_kbf           if rac = key
0191   490    ;       ifb$l_ebk           end of file block
0191   491    ;       ifb$l_mrn           max. record #
0191   492    ;       irb$l_rp            record #
0191   493    ;       irb$l_nrp           if rac = seq and irb$v_find_last = 0
0191   494    ;       irb$v_find          set if doing $find
0191   495    ;
0191   496    ;   output parameters:
0191   497    ;
0191   498    ;       r5          address of record in buffer
0191   499    ;       r4          bdb address, if any, else 0
0191   500    ;       r0          status code
0191   501    ;       rab$l_bkt           rrn of record accessed
0191   502    ;       r1-r3,r6,ap         destroyed
0191   503    ;
0191   504    ;   implicit outputs:
0191   505    ;
```

RM2GET
V04-001

J 5

RELATIVE SPECIFIC GET AND FIND   16-SEP-1984 01:03:37 VAX/VMS Macro V04-00   Page 14
GETREC2 - ROUTINE TO LOCATE RECORD IN BU   7-SEP-1984 17:13:37 [RMS.SRC]RM2GET.MAR;2   (13)

```
0191  506 ;        if error = eof, r2 has the requried hi vbn + 1
0191  507 ;
0191  508 ;    completion codes:
0191  509 ;
0191  510 ;        standard rms.
0191  511 ;
0191  512 ;    side effects:
0191  513 ;
0191  514 ;        process may have stalled waiting for access to the bucket.
0191  515 ;        bucket is left accessed.
0191  516 ;--
0191  517 ;
```

```
                        0191  519
                        0191  520  ;
                        0191  521  ;   entry point for $get and $find
                        0191  522  ;
                        0191  523
                        0191  524  GETREC2:
                        0191  525      $CSHFLAGS        <>                        ; need read only access to bkt
                        0193  526
                        0193  527  ;
                        0193  528  ;   alternate entry point for $put
                        0193  529  ;     (r3 already set to csh$m_lock)
                        0193  530  ;
                        0193  531
                        0193  532  RM$GETREC2_PUT::
              54  D4    0193  533      CLRL     R4                               ; zero bdb address
           20 A9  D4    0195  534      CLRL     IRB$L_CURBDB(R9)                 ; (both places that we look for it)
                        0198  535
                        0198  536  ; (flag for no bucket accessed)
                        0198  537  ;
                        0198  538  ;   get the record # to use based upon rac
                        0198  539  ;
                        0198  540  ;
                        0198  541
                        0198  542      ASSUME   RAB$C_SEQ EQ 0
                        0198  543      ASSUME   RAB$C_KEY EQ 1
                        0198  544      ASSUME   RAB$C_RFA EQ 2
                        0198  545      CASE     TYPE=B,SRC=RAB$B_RAC(R8),-
                        0198  546      DISPLIST=<SEQRAC,KEYRAC,RFARAC> ; dispatch to right routine
                        01A3  547
                        01A3  548  ;
                        01A3  549  ;   fall thru on bad rac value
                        01A3  550  ;
                        01A3  551  ERRRAC:
                        01A3  552
                        01A3  553      RMSERR   RAC                              ; bad rac value
              05  01A8  554      RSB
                        01A9  555
                        01A9  556  RFAERR:
                        01A9  557      RMSERR   RFA              ; rfa error (rfa=0)
              05  01AE  558      RSB
```

```
                              01AF      560
                              01AF      561 ;
                              01AF      562 ;    subroutine to get number of records per bucket into r0
                              01AF      563 ; outputs:
                              01AF      564 ;         r0 = number of records per bucket
                              01AF      565 ;         r2 = number of blocks per bucket
                              01AF      566 ;
                              01AF      567
                              01AF      568 RECS_BKT:
        52   5E AA   9A       01AF      569         MOVZBL    IFB$B_BKS(R10),R2        ; bucket size into r2
     50   52   09   78        01B3      570         ASHL      #9,R2,R0                 ; bytes/bucket now
        50   62 A9   A6       01B7      571         DIVW2     IRB$W_CSIZ(R9),R0        ; records/bucket
                    05        01BB      572         RSB
                              01BC      573
                              01BC      574 ;
                              01BC      575 ;    rac = rfa
                              01BC      576 ;
                              01BC      577 ;    set rp from rfa value in rab
                              01BC      578 ;
                              01BC      579
                              01BC      580 RFARAC:
     55   10 A8   D0          01BC      581         MOVL      RAB$L_RFA0(R8),R5        ; assume this really relative
             E7   13          01C0      582         BEQL      RFAERR                   ; zero is error
     62 53   00   E1          01C2      583         BBC       #CSH$V_LOCK,R3,ULKRP     ; ok if read function
             DB   11          01C6      584         BRB       ERRRAC                   ; rfa access error if put
                              01C8      585
                              01C8      586 ;
                              01C8      587
```

RM2GET   RELATIVE SPECIFIC GET AND FIND   16-SEP-1984 01:03:37   VAX/VMS Macro V04-00   Page 17
V04-001   GETREC2 - ROUTINE TO LOCATE RECORD IN BU   7-SEP-1984 17:13:37   [RMS.SRC]RM2GET.MAR;2   (19)

M 5

```
                  01C8      589
                  01C8      590  ;
                  01C8      591  ;   handle key buffer errors
                  01C8      592  ;
                  01C8      593  ;
                  01C8      594  ERRKSZ:
                  01C8      595          RMSERR  KSZ             ; invalid key length
             05   01CD      596          RSB
                  01CE      597
                  01CE      598  ERRKBF:
                  01CE      599          RMSERR  KBF             ; invalid key buffer
             05   01D3      600          RSB
                  01D4      601
                  01D4      602  ERRKEY:
                  01D4      603          RMSERR  KEY             ; key < or = 0
             05   01D9      604          RSB
                  01DA      605
                  01DA      606  ERRMRN:
        53   D5   01DA      607          TSTL    R3                      ; do cache flags indicate $put
        08   12   01DC      608          BNEQ    10$                     ; branch if yes (err = mrn)
                  01DE      609          ASSUME  RAB$C_SEQ EQ 0
      1E A8  95   01DE      610          TSTB    RAB$B_RAC(R8)           ; sequential access?
        03   12   01E1      611          BNEQ    10$                     ; branch if not
      00C0   31   01E3      612          BRW     ERREOF                  ; yes - give eof error
             05   01E6      613  10$:     RMSERR  MRN                     ; key > max. rec. #
                  01EB      614          RSB
                  01EC      615
```

```
                        01EC    617
                        01EC    618 ;
                        01EC    619 ;  rac = key
                        01EC    620 ;
                        01EC    621 ;  set rp from relative record number in key buffer
                        01EC    622 ;
                        01EC    623 ;
          34 A8   95    01EC    624 KEYRAC: TSTB    RAB$B_KSZ(R8)             ; zero key size?
             06   13    01EF    625         BEQL    10$                      ; branch if yes (default)
    04    34 A8   91    01F1    626         CMPB    RAB$B_KSZ(R8),#4         ; is it 4?
             D1   12    01F5    627         BNEQ    ERRKSZ                   ; branch if not
    55    30 B8   DE    01F7    628 10$:    MOVAL   @RAB$L_KBF(R8),R5       ; get key buffer addr
                        01FB    629         IFNORD  #4,(R5),ERRKBF         ; branch if not readable
       55  65   D0     0201    630         MOVL    (R5),R5                  ; pick up record #
    08 68  36   E0     0204    631         BBS     #RAB$V_KGT+ROP,(R8),KGT ; branch if kgt
    55    48 A9   D1    0208    632         CMPL    IRB$L_RP(R9),R5         ; same as current record?
             1A   12    020C    633         BNEQ    ULKRP                    ; nope, continue normally
             14   11    020E    634         BRB     CHKNLK                   ; check if no lock set
             55   D6    0210    635 KGT:    INCL    R5                       ; increment record #
             14   11    0212    636         BRB     ULKRP
                        0214    637
                        0214    638 ;
                        0214    639 ;  rac = seq
                        0214    640 ;
                        0214    641 ;  set rp from nrp unless doing a $get after a $find, in which case
                        0214    642 ;  the rp is correct as is
                        0214    643 ;
                        0214    644
                        0214    645 SEQRAC:
    55    40 A9   D0    0214    646         MOVL    IRB$L_NRP(R9),R5        ; assume next record
    0C 69  29   E0     0218    647         BBS     #IRB$V_FIND,(R9),ULKRP  ; branch if doing $find
    08 69  25   E1     021C    648         BBC     #IRB$V_FIND_LAST,(R9),ULKRP; or if last operation not $find
                        0220    649                                          ; (note: this bit will be clear
                        0220    650                                          ; for $put)
                        0220    651
    55    48 A9   D0    0220    652         MOVL    IRB$L_RP(R9),R5        ; re-get last record
    11 68  34   E1     0224    653 CHKNLK: BBC     #RAB$V_NLK+ROP,(R8),SETRP; don't unlock current record
                        0228    654                                          ; unless no lock desired on
                        0228    655                                          ; the new record
                        0228    656
                        0228    657 ;
                        0228    658 ;  if irb$v_unlock_rp set, unlock the current record
                        0228    659 ;
                        0228    660
    0D 69  2D   E5     0228    661 ULKRP:  BBCC    #IRB$V_UNLOCK_RP,(R9),SETRP; clear unlock flag and branch
                        022C    662                                          ; if auto unlock not req'd.
                        022C    663
    51    48 A9   D0    022C    664         MOVL    IRB$L_RP(R9),R1        ; get record #
             52   D4    0230    665         CLRL    R2                       ; clear hi word of rec #
             53   DD    0232    666         PUSHL   R3                       ; save cache flags
          FDC9'  30    0234    667         BSBW    RM$UNLOCK                ; unlock the record
                        0237    668                                          ; (ignore possible error)
             08   BA    0237    669         POPR    #^M<R3>                  ; restore cache flags
                        0239    670
                        0239    671 ;
                        0239    672 ;  set rp and check for validity
                        0239    673 ;  note: this is also an alternate entry point to get next record
```

RM2GET
V04-001

B 6

RELATIVE SPECIFIC GET AND FIND   16-SEP-1984 01:03:37 VAX/VMS Macro V04-00   Page 19
GETREC2 - ROUTINE TO LOCATE RECORD IN BU   7-SEP-1984 17:13:37 [RMS.SRC]RM2GET.MAR;2   (21)

```
                           0239    674 ;  specified by r5 (for sequential get and find after initially positioning
                           0239    675 ;  to a cell with no record)
                           0239    676 ;
                           0239    677
          48 A9   55   DO  0239    678 SETRP:   MOVL     R5,IRB$L_RP(R9)          ; save rec # in rp
                  95   15  023D    679          BLEQ     ERRKEY                   ; get out on bad rec #
          10 A8   55   DO  023F    680          MOVL     R5,RAB$W_RFA(R8)         ; set rfa from rec #
               14 A8  B4   0243    681          CLRW     RAB$W_RFA+4(R8)          ; be neat
          10 6A   38   EO  0246    682          BBS      #IFB$V_SEQFIL,(R10),15$  ; skip mrn check and don't
                           024A    683                                           ; return bkt if seq file
                           024A    684          ASSUME   RAB$C_SEQ EQ 0
          1E A8   95       024A    685          TSTB     RAB$B_RAC(R8)            ; sequential access?
                  04   12  024D    686          BNEQ     10$
       38 A8   55   DO      024F    687          MOVL     R5, RAB$L_BKT(R8)        ; only if sequential access
    55   00AC CA   D1  0253    688 10$:   CMPL     IFB$L_MRN(R10),R5        ; rec # within bounds?
                  80   19  0258    689          BLSS     ERRMRN                   ; branch if not
                           025A    690
                           025A    691 ;
                           025A    692 ;  calculate vbn and offset
                           025A    693 ;
                           025A    694
                  55   D7  025A    695 15$:   DECL     R5                       ; rec # - 1
                  FF50 30  025C    696          BSBW     RECS_BKT                 ; # records/bucket to r0
                           025F    697                                           ; loads r2 with bucket size
                           025F    698
                  56   D4  025F    699          CLRL     R6                       ; zero extend dividend
  4C A9   51   55   50  7B  0261    700          EDIV     R0,R5,R1,IRB$L_RP_OFF(R9); compute bkt # (in r1)
                           0267    701                                           ; and rec-in-bkt (in rp_off)
       4C A9   62 A9   A4  0267    702          MULW2    IRB$W_CSIZ(R9),IRB$L_RP_OFF(R9); compute offset in bucket
              51   52  C4  026C    703          MULL2    R2,R1                    ; get relative vbn
       51   00B0 CA   CO  026F    704          ADDL2    IFB$L_DVBN(R10),R1       ; and point past prolog
                  54   D5  0274    705          TSTL     R4                       ; already got buffer?
                  24   12  0276    706          BNEQ     SETOFF                   ; branch if yes
       44 A9   51   DO  0278    707          MOVL     R1,IRB$L_CURVBN(R9)      ; save vbn for later
                           027C    708
                           027C    709 RMSREADBKT2_UPD::
              52   51  CO  027C    710          ADDL2    R1,R2                    ; compute end vbn+1
       74 AA   52   D1  027F    711          CMPL     R2,IFB$L_EBK(R10)        ; past eof?
                  21   1A  0283    712          BGTRU    ERREOF                   ; branch if yes
                           0285    713
```

```
                            0285   715
                            0285   716  ;
                            0285   717  ;        entry point to read a bucket via rm$cache
                            0285   718  ;
                            0285   719  ;        inputs:
                            0285   720  ;
                            0285   721  ;                r8-r11              same as for rm$get2
                            0285   722  ;                r3                  cache flags
                            0285   723  ;                r1                  vbn
                            0285   724  ;                irb$l_rp_off        offset to record cell in bucket
                            0285   725  ;
                            0285   726  ;        outputs:
                            0285   727  ;
                            0285   728  ;                r5                  record address
                            0285   729  ;                r4                  bdb address (0 on failure)
                            0285   730  ;                r0                  status
                            0285   731  ;                r1-r3,ap            destroyed
                            0285   732  ;                irb$l_curbdb        bdb address
                            0285   733  ;
                            0285   734
                            0285   735  RM$READBKT2::
     52  5E AA  9A          0285   736          MOVZBL  IFB$B_BKS(R10),R2           ; bkt size to r2
  52  52  09   78           0289   737          ASHL    #9,R2,R2                    ; transfer size
                            028D   738          $CACHE  VBN=R1,SIZE=R2,FLAGS=R3,ERR=ERRCSH; access bkt
   20 A9  54   D0           0293   739          MOVL    R4,IRB$L_CURBDB(R9)         ; save bdb address
   55  4C A9   C0           0297   740          ADDL2   IRB$L_RP_OFF(R9),R5         ; add in record offset to buffer
                            029B   741                                              ; addr giving record addr
                            029B   742
                            029B   743          RSB
                     05     029B   743          RSB
                            029C   744
                            029C   745  ;
                            029C   746  ;        already have bdb.  compute new record buffer address.
                            029C   747  ;
                            029C   748
 55  18 A4  4C A9   C1      029C   749  SETOFF: ADDL3   IRB$L_RP_OFF(R9),BDB$L_ADDR(R4),R5
                            02A2   750          RMS$SUC                             ; show success
                     05     02A5   751          RSB
                            02A6   752
                            02A6   753  ;
                            02A6   754  ;        handle errors
                            02A6   755  ;
                            02A6   756
                            02A6   757  ERREOF: RMSERR  EOF                         ; say it's eof
                     05     02AB   758          RSB
                            02AC   759
               54    D4     02AC   760  ERRCSH: CLRL    R4                          ; show no bdb accessed
                     05     02AE   761          RSB
                            02AF   762
```

RM2GET   D 6
V04-001   RELATIVE SPECIFIC GET AND FIND 16-SEP-1984 01:03:37 VAX/VMS Macro V04-00 Page 21
         GETFIND2 - COMMON $GET AND $FIND CODE TO 7-SEP-1984 17:13:37 [RMS.SRC]RM2GET.MAR;2 (24)

```
02AF   764              .SBTTL  GETFIND2 - COMMON $GET AND $FIND CODE TO ACCESSRECORD
02AF   765
02AF   766    ;++
02AF   767    ;
02AF   768    ;   this routine performs the following functions:
02AF   769    ;
02AF   770    ;       1. checks r0 status code and if in error checks for eof.
02AF   771    ;          if eof and rac is not sequential, changes the status to rnf
02AF   772    ;          (record not found) unless the nxr rop bit is set, in which case
02AF   773    ;          it changes the status to ok_rnf.
02AF   774    ;
02AF   775    ;       2. if r0 does not indicate an error, checks the control byte of the
02AF   776    ;          record to see if record exists.  if not and rac not = seq,
02AF   777    ;          returns rnf (del if rac=rfa) unless the nxr rop bit is set, in which
02AF   778    ;          case it returns either ok_rnf or ok_del.  if rac = seq, non-existent
02AF   779    ;          records are skipped until either a valid record is found or eof
02AF   780    ;          is encountered.
02AF   781    ;
02AF   782    ;   inputs:
02AF   783    ;
02AF   784    ;       r0              status code
02AF   785    ;       r4              bdb address if one, else 0
02AF   786    ;       r5              record cell address
02AF   787    ;       r8-r11          same as for rm$get2
02AF   788    ;       irb$l_rp        current record #
02AF   789    ;
02AF   790    ;   outputs:
02AF   791    ;
02AF   792    ;       r7              status code
02AF   793    ;       r5              record cell address + 1 (i.e., past control byte)
02AF   794    ;       r0-r3,r6        destroyed
02AF   795    ;       ap              if success (low bit set r7), ap = 0 if irb$v_unlock_rp
02AF   796    ;                       only to determine whether rp needs to be unlocked on
02AF   797    ;                       errors detected later.  ap = 1 if rp is to be unlocked
02AF   798    ;                       on later errors regardless of irb$v_unlock_rp
02AF   799    ;--
02AF   800    ;
```

```
                      02AF      802  GETFIND2:
       57  50  D0     02AF      803          MOVL    R0,R7                      ; save status code
          7F  50  E9  02B2      804          BLBC    R0,CHKEOF                  ; branch on error
       48 6A  38  E0  02B5      805          BBS     #IFBSV_SEQFIL,(R10),LOCK   ; skip this junk if seq file
                      02B9      806          ASSUME  FAB$C_SEQ EQ 0
          1E A8  95   02B9      807          TSTB    RAB$B_RAC(R8)              ; sequential access mode?
              44  13  02BC      808          BEQL    SEQACC                     ; branch if yes
       01  1E A8  91  02BE      809          CMPB    RAB$B_RAC(R8),#RAB$C_KEY   ; is it key access?
              0A  12  02C2      810          BNEQ    2$                         ; branch if not
    04 A8  00600000 8F D3 02C4  811          BITL    #RAB$M_KGE!RAB$M_KGT,RAB$L_ROP(R8) ; is kge or kgt set?
              34  12  02CC      812          BNEQ    SEQACC                     ; branch if yes
          08  85  91  02CE      813  2$:      CMPB    (R5)+,#DLC$M_REC          ; does record exist?
              2E  13  02D1      814          BEQL    LOCK                       ; branch if yes
    FF A5  F3 8F  93  02D3      815          BITB    #^C <DLC$M_DELETED!DLC$M_REC>,-1(R5) ; valid bit combination?
              77  12  02D8      816          BNEQ    ERRIRC_BR                  ; branch if not
       12 68  37  E0  02DA      817          BBS     #RAB$V_NXR+ROP,(R8),RTNNXR ; branch if user wants the
                      02DE      818                                            ; non-existent record
       02  1E A8  91  02DE      819          CMPB    RAB$B_RAC(R8),#RAB$C_RFA   ; is rac=rfa?
              06  12  02E2      820          BNEQ    ERRRNF                     ; branch if not (err = rnf)
                      02E4      821          RMSERR  DEL,R7                     ; set error code
              05     02E9      822          RSB                                ; return
              05     02EA      823  ERRRNF:  RMSERR  RNF,R7
              05     02EF      824          RSB
                      02F0      825
                      02F0      826
                      02F0      827  RTNNXR:
    07 FF A5  02  E1  02F0      828          BBC     #DLC$V_DELETED,-1(R5),OK_RNF ; branch if record not deleted
          05  11     02F5      829          RMSSUC  OK_DEL,R7                  ; indicate read of deleted record
                      02FA      830          BRB     LOCK                       ; and continue
                      02FC      831
                      02FC      832  OK_RNF:  RMSSUC  OK_RNF,R7                 ; indicate read of non-ex rec.
                      0301      833
              05     0301      834  LOCK:    RSB                                ; and continue
                      0302      835
                      0302      836
                      0302      837  ; handle sequential access
                      0302      838  ; if record deleted or never existed try next record.
                      0302      839  ;
                      0302      840
          08  85  91  0302      841  SEQACC:  CMPB    (R5)+,#DLC$M_REC          ; does record exist?
              FA  13  0305      842          BEQL    LOCK                       ; branch if yes
    FF A5  F3 8F  93  0307      843          BITB    #^C <DLC$M_DELETED!DLC$M_REC>,-1(R5) ; valid bit combination?
              43  12  030C      844          BNEQ    ERRIRC_BR                  ; branch if not
    50  62 A9  01  78 030E      845          ASHL    #1,IRB$W_CSIZ(R9),R0       ; get twice the cell size
       50  4C A9  A0  0313      846          ADDW2   IRB$W_RP_OFF(R9),R0        ; plus the record offset
                      0317      847                                            ; (i.e. the end of the next rec)
       50  16 A4  B1  0317      848          CMPW    BDB$W_SIZE(R4),R0          ; next record in this bkt?
              0A  1E  031B      849          BGEQU   10$                        ; branch if yes (omit release)
              53  D4  031D      850          CLRL    R3                         ; no options wanted
          FCDE' 30  031F      851          BSBW    RMS$RELEASE                ; release access to bucket
              54  D4  0322      852          CLRL    R4                         ; show no bdb
       20 A9  D4     0324      853          CLRL    IRB$L_CURBDB(R9)           ; and no current bdb
    55  01  48 A9 C1 0327      854  10$:     ADDL3   IRB$L_RP(R9),#1,R5        ; get next record #
              53  D4  032C      855          CLRL    R3                         ; indicate get
          FF08  30  032E      856          BSBW    SETRP                      ; get the record
          FF7B  31  0331      857          BRW     GETFIND2                   ; and check it out
```

```
                        0334    859
                        0334    860  ;
                        0334    861  ; got an error.
                        0334    862  ;
                        0334    863  ; if error = eof perform following:
                        0334    864  ;
                        0334    865  ;     if rac not = seq, change error code to record not found, unless
                        0334    866  ;     user is reading non-existent records, in which case set status to
                        0334    867  ;     ok_rnf and continue
                        0334    868  ;
                        0334    869
                        0334    870  CHKEOF:
  827A BF   50   B1     0334    871          CMPW    R0,#RMS$_EOF&^XFFFF          ; was error = eof?
           12   12      0339    872          BNEQ    GF2XT1                       ; branch if not
                        033B    873          ASSUME  RAB$C_SEQ EQ 0
        1E A8   95      033B    874          TSTB    RAB$B_RAC(R8)                ; rac = seq?
           0D   13      033E    875          BEQL    GF2XTT                       ; branch if yes
  0A 68   37   E0       0340    876          BBS     #RAB$V_NXR+ROP,(R8),OK_RNF1; modify status and continue
  05 6A   38   E0       0344    877          BBS     #IFB$V_SEQFIL,(R10),GF2XT1; eof if really seq file
                        0348    878          RMSERR  RNF,R7                       ; set code to rec. not found
           05           034D    879  GF2XT1: RSB
                        034E    880
                        034E    881  OK_RNF1:
        FFAB   31       034E    882          BRW     OK_RNF                       ; extended branch
                        0351    883
                        0351    884  ERRIRC_BR:
        FD6B   31       0351    885          BRW     ERRIRC                       ; extended branch
                        0354    886
```

```
                        0354    888                 .SBTTL  GETLOCK2 - LOCK RELATIVE RECORD IF NECESSARY
                        0354    889
                        0354    890       ;++
                        0354    891       ;
                        0354    892       ;  GETLOCK2
                        0354    893       ;
                        0354    894       ;  if record locking not required, return to caller.
                        0354    895       ;  otherwise, if the file is write accessed and the nlk (no lock) rop
                        0354    896       ;  bit is clear, lock the record.  if the file is either not write accessed or
                        0354    897       ;  nlk is set, need merely check that no other user has record locked.
                        0354    898       ;
                        0354    899       ;  however, if file is not write-accessed, but user wants to lock for read,
                        0354    900       ;  allow him to.
                        0354    901       ;
                        0354    902       ;  inputs:
                        0354    903       ;
                        0354    904       ;       r0              status code
                        0354    905       ;       r4              bdb address if one, else 0
                        0354    906       ;       r5              record cell address
                        0354    907       ;       r8-r11          same as for rm$get2
                        0354    908       ;       irb$l_rp        current record number
                        0354    909       ;
                        0354    910       ;  outputs:
                        0354    911       ;
                        0354    912       ;       r7              status code
                        0354    913       ;       r4              may be loaded with contents of irb$l_curbdb
                        0354    914       ;
                        0354    915       ;  side effects
                        0354    916       ;
                        0354    917       ;       record locked
                        0354    918       ;
                        0354    919       ;--
                        0354    920
                        0354    921       GETLOCK2:
        6A    33  E0    0354    922                 BBS     #IFB$V_NORECLK,(R10),-
              74        0357    923                                 GF2XIT          ; branch if no locking
  51   48  A9  D0       0358    924                 MOVL    IRB$L_RP(R9),R1         ; set rec #
              52  D4    035C    925                 CLRL    R2                      ; and high half
              34  E0    035E    926                 BBS     #RAB$V_NLK+ROP,-
        58  68  E0      0360    927                                 (R8),Q LOCK     ; branch if lock not wanted
  04 6A    30  E0       0362    928                 BBS     #IFB$V_WRTACC,(R10),10$ ; branch if write accessed
              22  E1    0366    929                 BBC     #RAB$V_REA+ROP,-
        50  68          0368    930                                 (R8),Q LOCK     ; branch if not locking for read
        FC93' 30        036A    931       10$:      BSBW    RM$LOCK                 ; lock record
        55 50 E9        036D    932                 BLBC    R0,SETR7                ; branch if failure
              54 D5     0370    933                 TSTL    R4                      ; is a bucket accessed?
              25 13     0372    934                 BEQL    CHKEOF1                 ; branch if none
  8061 8F  50 B1        0374    935                 CMPW    R0,#RMS$_OK_WAT&^XFFFF  ; did we lock record only after wait?
              06 12     0379    936                 BNEQ    CHKULK                  ; if neq no, don't reaccess bucket
        FE13  30        037B    937                 BSBW    GETREC2                 ; reaccess bucket that STALL deaccessed
        FF2E  30        037E    938                 BSBW    GETFIND2                ; and reaccess record
                        0381    939       CHKULK:
              5C D4     0381    940                 CLRL    AP                      ; initialize
              2D E4     0383    941                 BBSC    #IRB$V_UNLOCK_RP,-      ; if already set, this means
        07 69          0385    942                                 (R9),10$         ; auto locked record not unlockd
                        0387    943
  8039 8F  50 B1        0387    944                 CMPW    R0,#RMS$_OK_ALK&^XFFFF  ; was record already locked
```

```
           32  13  038C   945            BEQL    CHKR7                           ; yes, don't unlock automatically
        5C 01  D0  038E   946  10$:      MOVL    #1,AP                           ; must unlock record if further
                   0391   947                                                    ; errors encountered
                   0391   948
     2B 68 32  E0  0391   949            BBS     #RAB$V_ULK+ROP,(R8),CHKR7; leave unlock_rp clear if
                   0395   950                                                    ; manual lock
     27 69 2D  E3  0395   951            BBCS    #IRB$V_UNLOCK_RP,(R9),CHKR7; indicate unlock required
                   0399   952                                                    ; and branch
                   0399   953  ;
                   0399   954  ; we have just locked a record but there is no bucket accessed.
                   0399   955  ; could only be a lock on a non-existent record past eof. check that
                   0399   956  ; record is still past eof to avoid returning a false status of ok_rnf
                   0399   957  ; if record has been added to file since we last checked.
                   0399   958  ;
                   0399   959
                   0399   960  CHKEOF1:
     53 5E AA  9A  0399   961            MOVZBL  IFB$B_BKS(R10),R3               ; get # blks/bkt
     53 44 A9  C0  039D   962            ADDL2   IRB$L_CURVBN(R9),R3            ; + current vbn
     53 74 AA  D1  03A1   963            CMPL    IFB$L_EBK(R10),R3              ; still past eof?
        02  1E  03A5   964            BGEQU   10$                            ; branch if not
        D8  11  03A7   965            BRB     CHKULK                         ; continue
     FC54' 30  03A9   966  10$:      BSBW    RM$UNLOCK                      ; unlock the record
     51 44 A9  D0  03AC   967            MOVL    IRB$L_CURVBN(R9),R1           ; restore vbn
                   03B0   968            $CSHFLAGS       <>                     ; no need to lock
     FED0  30  03B2   969            BSBW    RM$READBKT2                   ; go read the bucket
     FEF7  30  03B5   970            BSBW    GETFIND2                      ; and check it out
        9A  11  03B8   971            BRB     GETLOCK2                      ; lock it again!
                   03BA   972
                   03BA   973  ;
                   03BA   974  ; either not write accessed or nlk set.
                   03BA   975  ; need merely check that record is not locked from readers.
                   03BA   976  ;
                   03BA   977
     FC43' 30  03BA   978  QLOCK:    BSBW    RM$QUERY_LCK                  ; check if read ok
        05 50 E9  03BD   979            BLBC    R0,SETR7                      ; branch on error
                   03C0   980
                   03C0   981  ;
                   03C0   982  ; update status in r7 with the result of lock or querylock unless r7
                   03C0   983  ; already has some status other than rms$_normal
                   03C0   984  ;
                   03C0   985
        01 57 B1  03C0   986  CHKR7:    CMPW    R7,#RMS$_NORMAL&^XFFFF         ; r7 = normal?
        07  12  03C3   987            BNEQ    GF2XIT                        ; branch if not
     54 20 A9  D0  03C5   988  SETR7:    MOVL    IRB$L_CURBDB(R9),R4          ; Reload R4 with curbdb in case
                   03C9   989                                                    ; a record lock error has left it
                   03C9   990                                                    ; not accessed.
        57 50 D0  03C9   991            MOVL    R0,R7                         ; update status
        05  03CC   992  GF2XIT:   RSB
                   03CD   993
                   03CD   994            .END
```

Symbol	Value			Symbol	Value		
$$.PSECT_EP	= 00000000			IRB$L_CURBDB	= 00000020		
$$.TMP	= 00000000			IRB$L_CURVBN	= 00000044		
$$RMSTEST	= 0000001A			IRB$L_NRP	= 00000040		
$$RMS_PBUGCHK	= 00000010			IRB$L_RP	= 00000048		
$$RMS_TBUGCHK	= 00000008			IRB$L_RP_OFF	= 0000004C		
$$RMS_UMODE	= 00000004			IRB$V_FIND	= 00000029		
BDB$B_FLGS	= 0000000A			IRB$V_FIND_LAST	= 00000025		
BDB$L_ADDR	= 00000018			IRB$V_UNLOCK_RP	= 0000002D		
BDB$V_NOLOCATE	= 00000004			IRB$W_CSIZ	= 00000062		
BDB$W_SIZE	= 00000016			IRB$W_RP_OFF	= 0000004C		
CHKEOF	00000334	R	01	KEYRAC	000001EC	R	01
CHKEOF1	00000399	R	01	KGT	00000210	R	01
CHKLOC	000000DD	R	01	LOCK	00000301	R	01
CHKNLK	00000224	R	01	LONG_PROBE	0000013D	R	01
CHKR7	000003C0	R	01	MOVE_MODE	000000F1	R	01
CHKULK	00000381	R	01	MOVREC	0000010E	R	01
CLEANUP	00000000	R	01	NTRETF	00000180	R	01
CLN1BR	000000B5	R	01	NTRETG	0000003C	R	01
CSH$V_LOCK	= 00000000			NULL_REC	00000090	R	01
DLC$M_DELETED	= 00000004			OK_RNF	000002FC	R	01
DLC$M_REC	= 00000008			OK_RNF1	0000034E	R	01
DLC$V_DELETED	= 00000002			PIO$A_TRACE	********	X	01
ERRCSA	000002AC	R	01	PROBEB	000000FC	R	01
ERREOF	000002A6	R	01	QLOCK	000003BA	R	01
ERRIRC	000000BF	R	01	RAB$B_KSZ	= 00000034		
ERRIRC_BR	00000351	R	01	RAB$B_RAC	= 0000001E		
ERRKBF	000001CE	R	01	RAB$C_KEY	= 00000001		
ERRKEY	000001D4	R	01	RAB$C_RFA	= 00000002		
ERRKSZ	000001C8	R	01	RAB$C_SEQ	= 00000000		
ERRMRN	000001DA	R	01	RAB$L_BKT	= 00000038		
ERRRAC	000001A3	R	01	RAB$L_KBF	= 00000030		
ERRRHB	000000B0	R	01	RAB$L_RBF	= 00000028		
ERRRNF	000002EA	R	01	RAB$L_RFA0	= 00000010		
ERRRTB	00000160	R	01	RAB$L_RHB	= 0000002C		
ERRUBF	000000D2	R	01	RAB$L_ROP	= 00000004		
ERRUSZ	000000CB	R	01	RAB$L_STV	= 0000000C		
FAB$C_FIX	= 00000001			RAB$L_UBF	= 00000024		
FAB$C_SEQ	= 00000000			RAB$M_KGE	= 00200000		
FAB$C_VFC	= 00000003			RAB$M_KGT	= 00400000		
FAB$V_UPD	= 00000003			RAB$V_KGT	= 00000016		
GETFIND	000002AF	R	01	RAB$V_LOC	= 00000010		
GETLOCK2	00000354	R	01	RAB$V_NLK	= 00000014		
GETREC2	00000191	R	01	RAB$V_NXR	= 00000017		
GF2XIT	000003CC	R	01	RAB$V_REA	= 00000002		
GF2XT1	0000034D	R	01	RAB$V_ULK	= 00000012		
IFB$B_BKS	= 0000005E			RAB$W_RFA	= 00000010		
IFB$B_FAC	= 00000022			RAB$W_RSZ	= 00000022		
IFB$B_FSZ	= 0000005F			RAB$W_USZ	= 00000020		
IFB$B_RFMORG	= 00000050			RECS_BKT	000001AF	R	01
IFB$L_DVBN	= 000000B0			RFAERR	000001A9	R	01
IFB$L_EBK	= 00000074			RFARAC	000001BC	R	01
IFB$L_MRN	= 000000AC			RM$CACHE	********	X	01
IFB$V_NORECLK	= 00000033			RM$CLN2_DEL	00000011	RG	01
IFB$V_SEQFIL	= 00000038			RM$CLN2_PUT	00000003	RG	01
IFB$V_WRTACC	= 00000030			RM$CLN2_UPD	0000000B	RG	01
IFB$W_MRS	= 00000060			RM$EXRMS	********	X	01
IRB$B_MODE	= 0000000A			RM$FIND2	0000016F	RG	01

```
RM$GET2                              0000002A RG     01
RM$GETREC2_PUT                       00000193 RG     01
RM$LOCK                              ********   X    01
RM$QUERY_LCK                         ********   X    01
RM$READBKT2                          00000285 RG     01
RM$READBKT2_UPD                      0000027C RG     01
RM$RELEASE                           ********   X    01
RM$RLS2                              00000124 RG     01
RM$UNLOCK                            ********   X    01
RMS$_DEL                          =  00018262
RMS$_EOF                          =  0001827A
RMS$_IRC                          =  0001857C
RMS$_KBF                          =  0001858C
RMS$_KEY                          =  00018594
RMS$_KSZ                          =  000185A4
RMS$_MRN                          =  000185CC
RMS$_NORMAL                       =  00010001
RMS$_OK_ALK                       =  00018039
RMS$_OK_DEL                       =  00018041
RMS$_OK_RNF                       =  00018049
RMS$_OK_WAT                       =  00018061
RMS$_RAC                          =  00018644
RMS$_RFA                          =  0001865C
RMS$_RHB                          =  0001866C
RMS$_RNF                          =  000182B2
RMS$_RTB                          =  000181A8
RMS$_UBF                          =  000186EC
RMS$_USZ                          =  000186F4
ROP                               =  00000020
RSZFIX                               000000D9 R      01
RTNNXR                               000002F0 R      01
SEQACC                               00000302 R      01
SEQRAC                               00000214 R      01
SETNRP                               0000011E R      01
SETOFF                               0000029C R      01
SETR7                                000003C5 R      01
SETRP                                00000239 R      01
SETRSZ                               0000011A R      01
TPT$L_FIND2                          ********   X    01
TPT$L_GET2                           ********   X    01
ULKRP                                00000228 R      01
VFCREC                               00000064 R      01
```

+-----------------+
! Psect synopsis !
+-----------------+

PSECT name	Allocation		PSECT No.		Attributes										
. ABS .	00000000	(0.)	00	(0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE
RM$RMS2	000003CD	(973.)	01	(1.)	PIC	USR	CON	REL	GBL	NOSHR	EXE	RD	NOWRT	NOVEC	BYTE
ABS	00000000	(0.)	02	(2.)	NOPIC	USR	CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE

```
                                    +----------------------------+
                                    ! Performance indicators !
                                    +----------------------------+

Phase                   Page faults   CPU Time      Elapsed Time
-----                   -----------   --------      ------------
Initialization                   32   00:00:00.08   00:00:00.86
Command processing              114   00:00:00.77   00:00:03.13
Pass 1                          328   00:00:11.25   00:00:23.98
Symbol table sort                 0   00:00:01.31   00:00:01.57
Pass 2                          175   00:00:03.15   00:00:08.94
Symbol table output              19   00:00:00.15   00:00:01.08
Psect synopsis output             1   00:00:00.02   00:00:00.02
Cross-reference output            0   00:00:00.00   00:00:00.00
Assembler run totals            671   00:00:16.73   00:00:39.60
```

The working set limit was 1500 pages.
63172 bytes (124 pages) of virtual memory were used to buffer the intermediate code.
There were 50 pages of symbol table space allocated to hold 1021 non-local and 28 local symbols.
994 source lines were read in Pass 1, producing 15 object records in Pass 2.
29 pages of virtual memory were used to define 28 macros.

```
                                    +------------------------------+
                                    ! Macro library statistics !
                                    +------------------------------+

Macro library name                          Macros defined
------------------                          --------------
_$255$DUA28:[RMS.OBJ]RMS.MLB;1                     17
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                      3
_$255$DUA28:[SYSLIB]STARLET.MLB;2                   4
TOTALS (all libraries)                             24
```
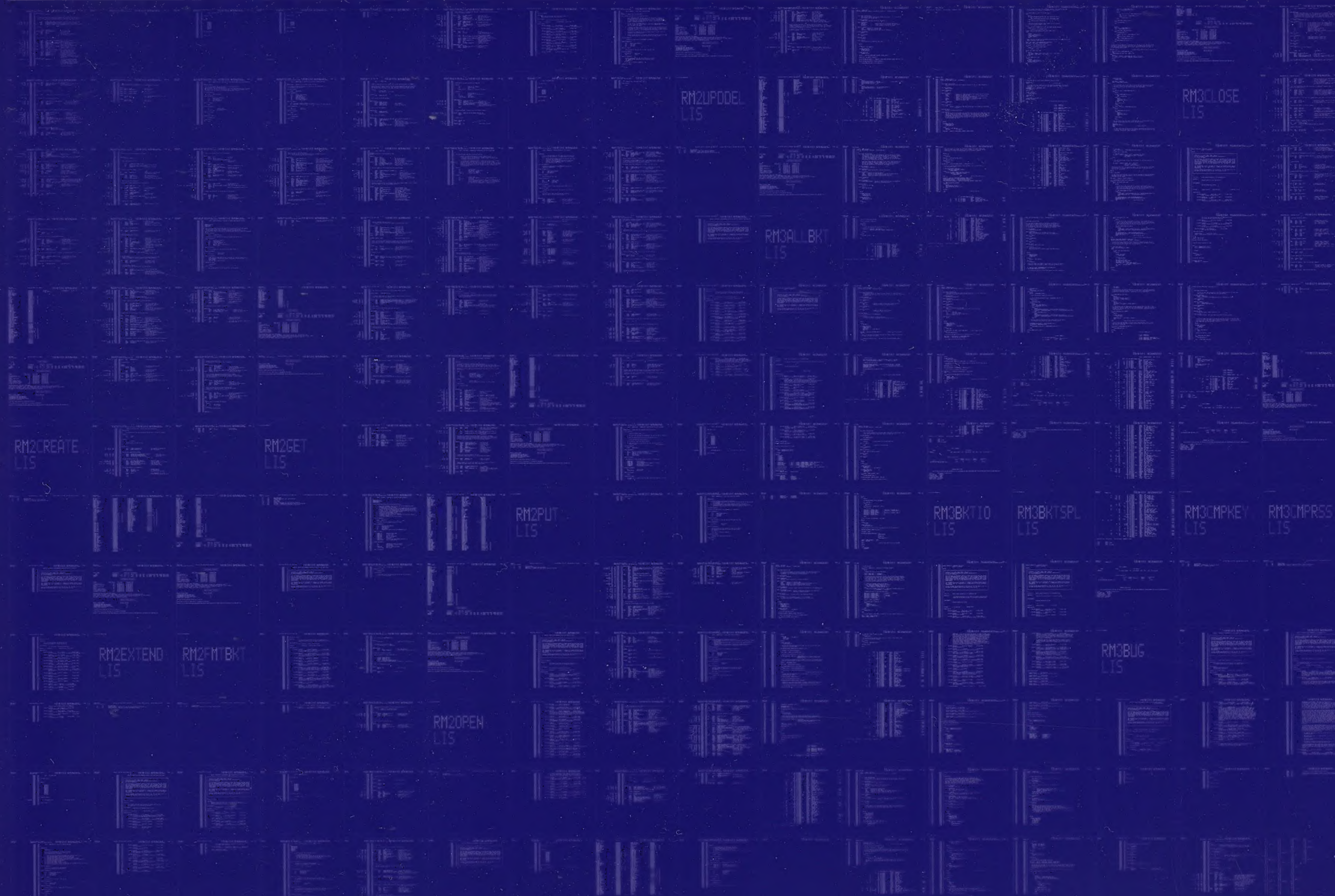
1137 GETS were required to define 24 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:RM2GET/OBJ=OBJ$:RM2GET MSRC$:RM2GET/UPDATE=(ENH$:RM2GET)+EXECML$/LIB+LIB$:RMS/LIB